

README: Final Project Part 3

Names: Mona Abualya, Malek Abualya, Emma, Sturm

GitHub repo:

<https://github.com/malekabualya1109/thoughtstreamcs483>

GitHub Commits:

<https://github.com/malekabualya1109/thoughtstreamcs483/commits/main/>

SET UP INSTRUCTIONS:

STEP 1 – MUST INSTALL THE FOLLOWING PACKAGES FOR FRONTEND DISPLAY:

npm install @lottiefles/react-lottie-player

npm install lottie-web

npm install lottie-web react-lottie

npm install @lottiefles/dotlottie-react

npm install --save @fortawesome/react-fontawesome @fortawesome/free-regular-svg-icons @fortawesome/free-solid-svg-icons

STEP 2 – MUST PUT A PROXY IN VITE.CONFIG.JS

This is how the vite.config.js file looks like:

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [react()],
  server: {
    port: 5173,
    proxy: {
      '/api': 'http://localhost:5000'
    }
  }
});
```

STEP 3 – ENSURE BACKEND AND FRONTEND ENV FILES ARE ACCURATE

FRONT END:

VITE_API_BASE_URL=http://localhost:5000/api

VITE_WEATHER_API=https://api.openweathermap.org/data/2.5/weather

WEATHER_FORECAST_API=https://api.openweathermap.org/data/2.5/forecast

VITE_WEATHER_API_KEY=<api weather key>

VITE_GOOGLE_CLIENT_ID=<google client id>

GOOGLE_CLIENT_SECRET=<google secret>

SESSION_SECRET=<secret>

BACKEND:

MONGO_URI=mongodb+srv://<username>:<pass>@diarycluster.vpiitjf.mongodb.net/diarydb?retryWrites=true&w=majority&appName=diaryCluster

PORT=5000

WEATHER_API=<weather-api key>

WEATHER_FORECAST_API=https://api.openweathermap.org/data/2.5/forecast

GOOGLE_CLIENT_ID=<google client id>

GOOGLE_CLIENT_SECRET=<google secret>

GOOGLE_CALLBACK_URL=http://localhost:5000/auth/google/callback

SESSION_SECRET=<google session secret>

JWT_SECRET=<random secret>

STEP 4 - RUN BACKEND

npm run dev

STEP 5 – RUN FRONTEND

npm run dev

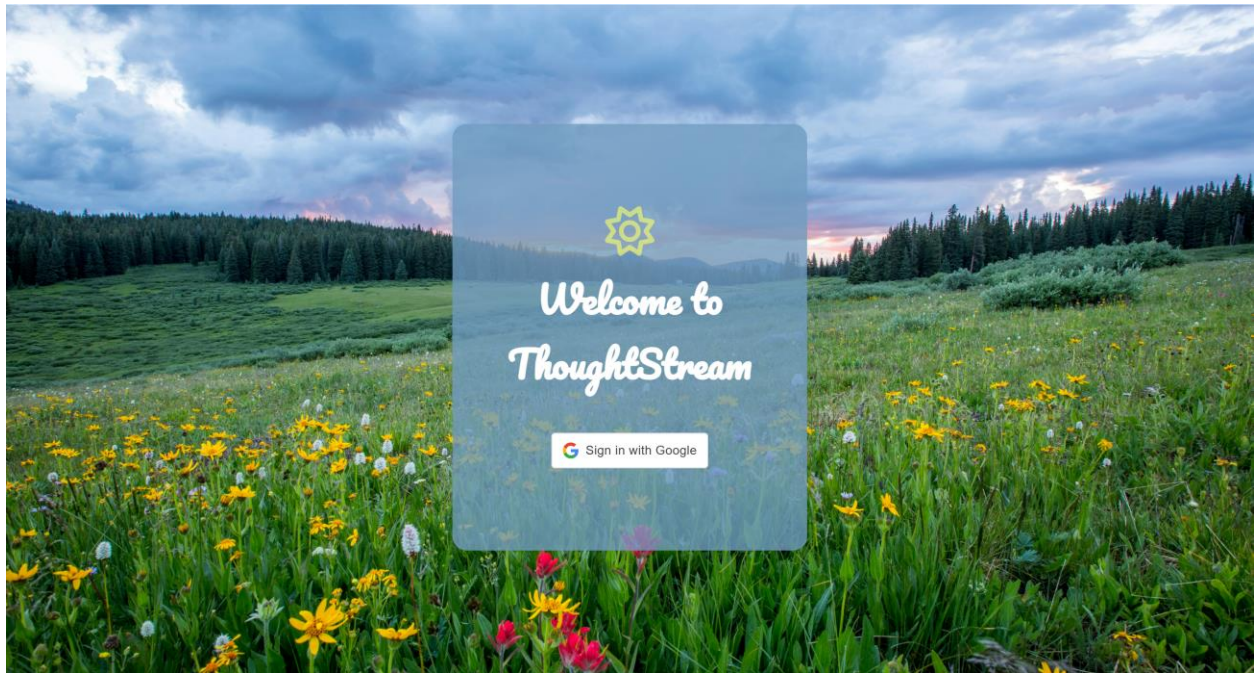
- copy and paste the website local url into a browser -> <http://localhost:5173/>
- Once loaded, should see the login screen

EXTRA CREDIT – validation.js was done for extra credit

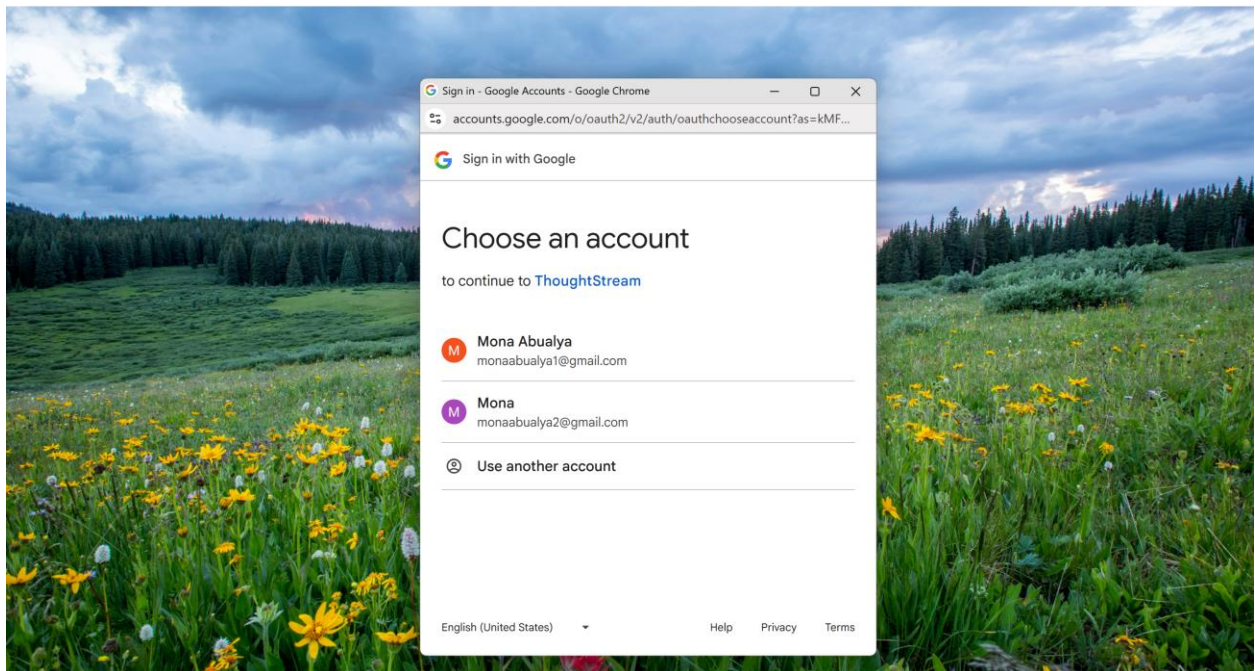
- Title, content, and location are required
- The rest of the inputs are optional
- Input for title and content must be greater than 3 characters but less than 100 characters

SCREENSHOTS DEMONSTRATING FLOW:

Login page:

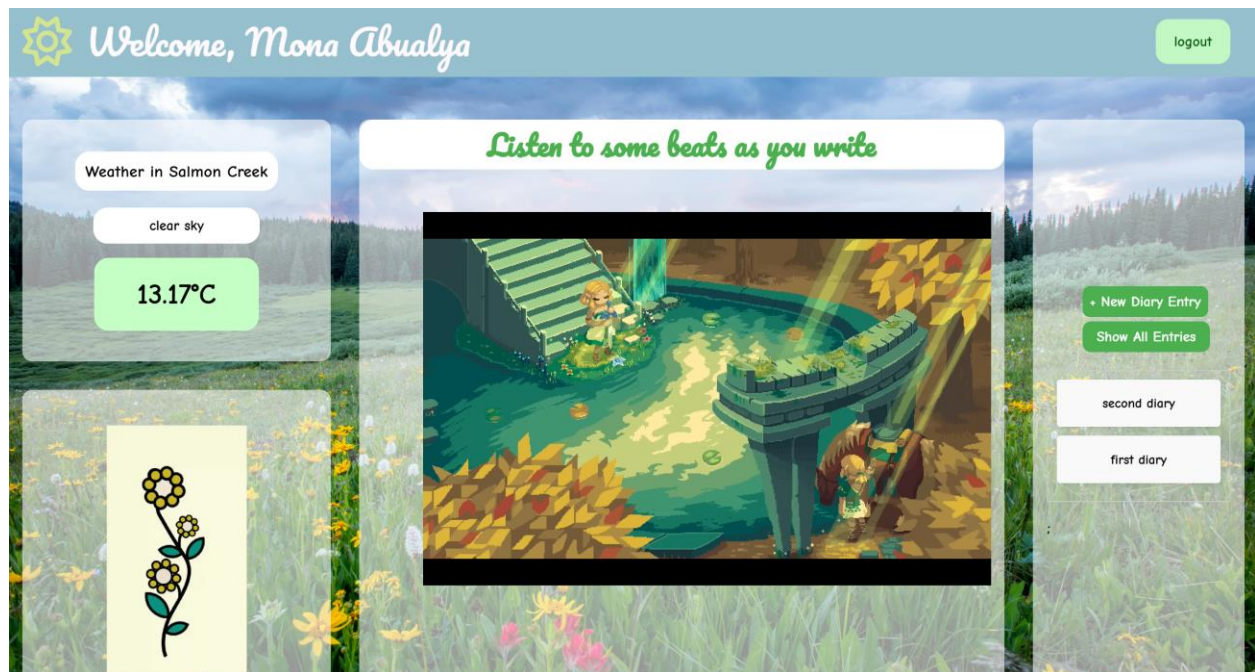
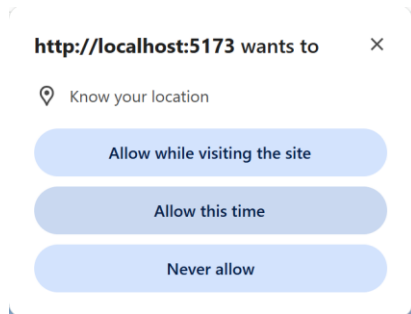


Google authentication page:



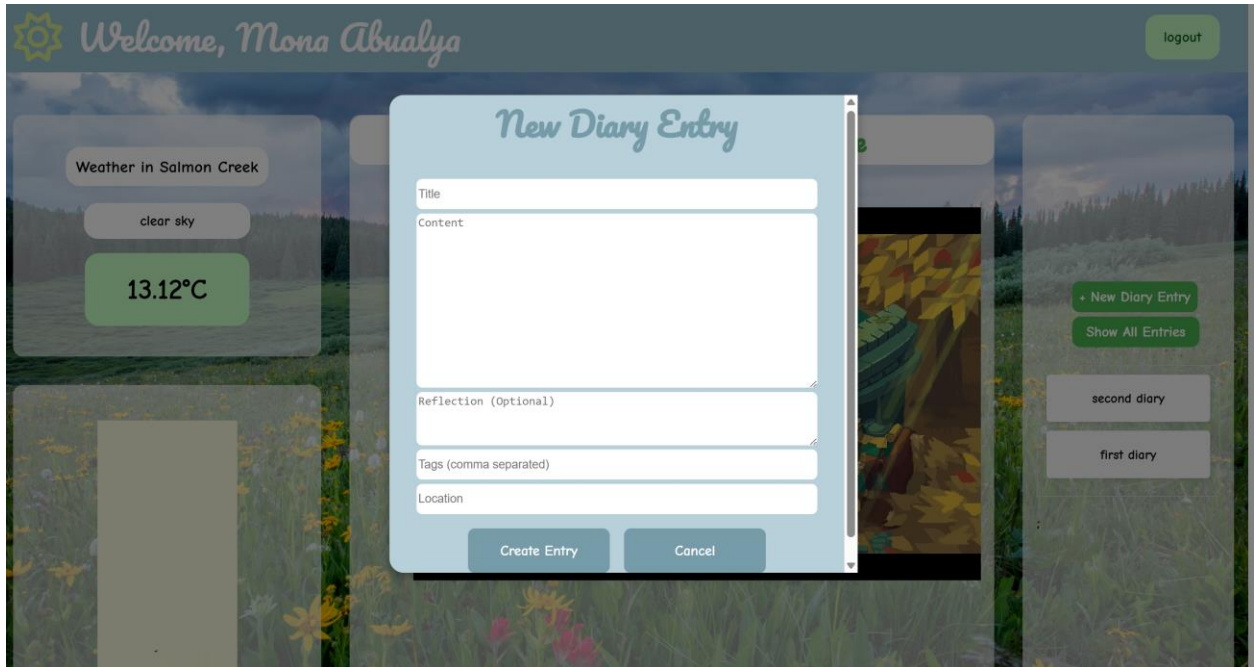
Dashboard page:

- ENABLE LOCATION TO FETCH WEATHER DATA FOR CURRENT LOCATION

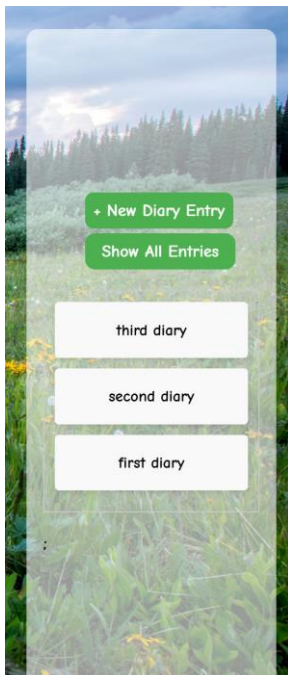


Adding a new entry:

- Click on the “+ New Diary Entry”
- Title, Content, location is required – must be at least 4 characters long and shorter than 100 characters



- Entry title displayed on the right side of screen

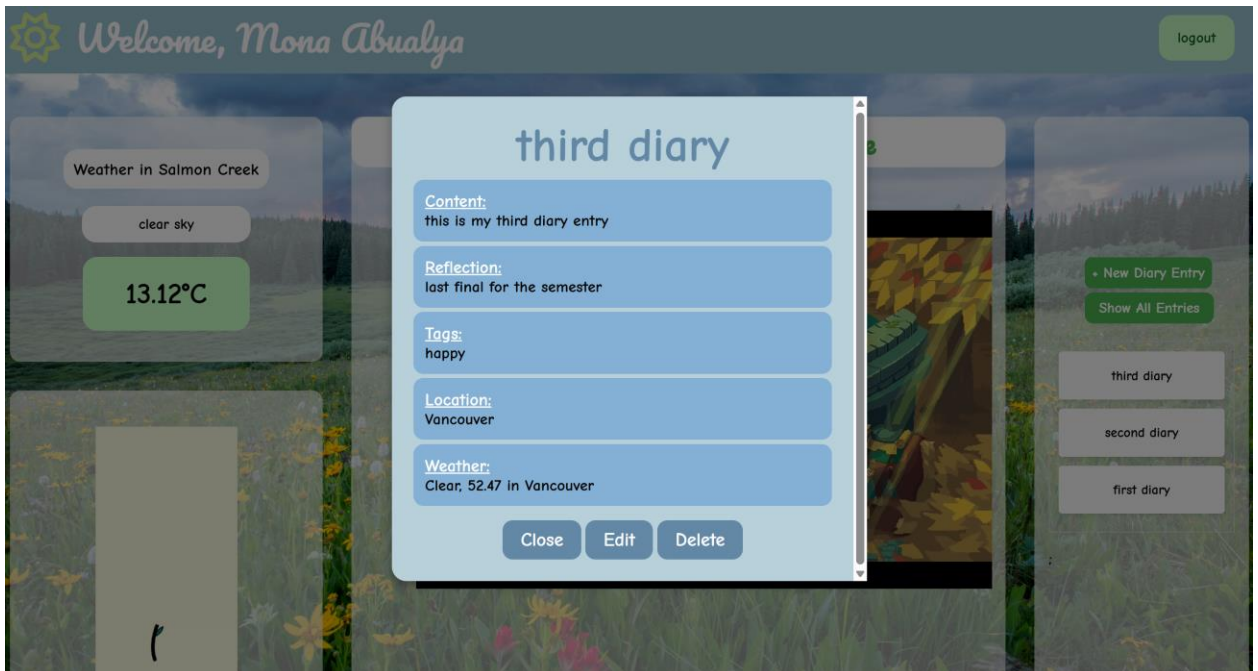


Viewing a particular entry:

- Click on the entry you want to view from the right side of screen (will be hovered)

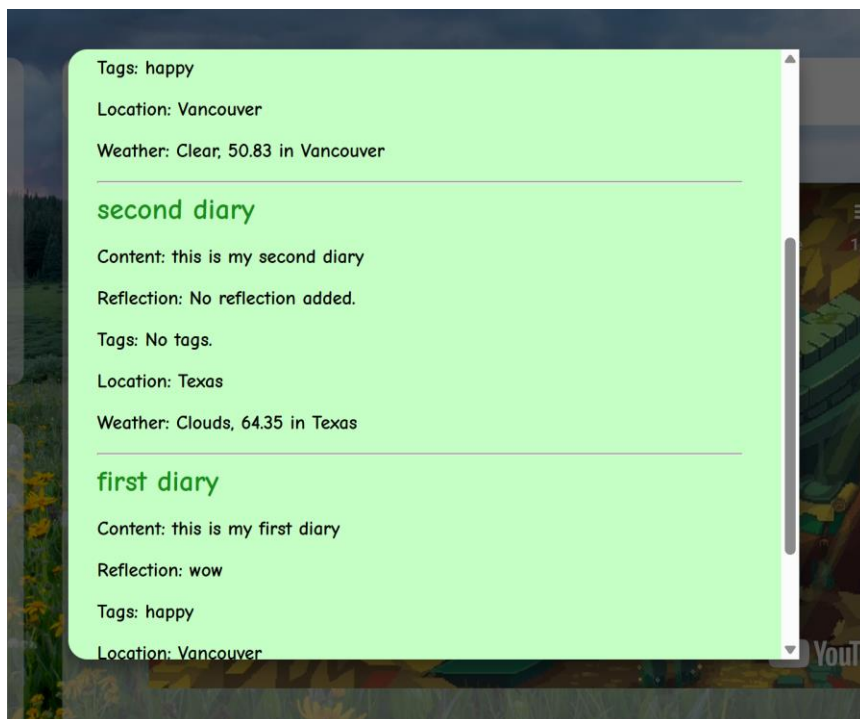
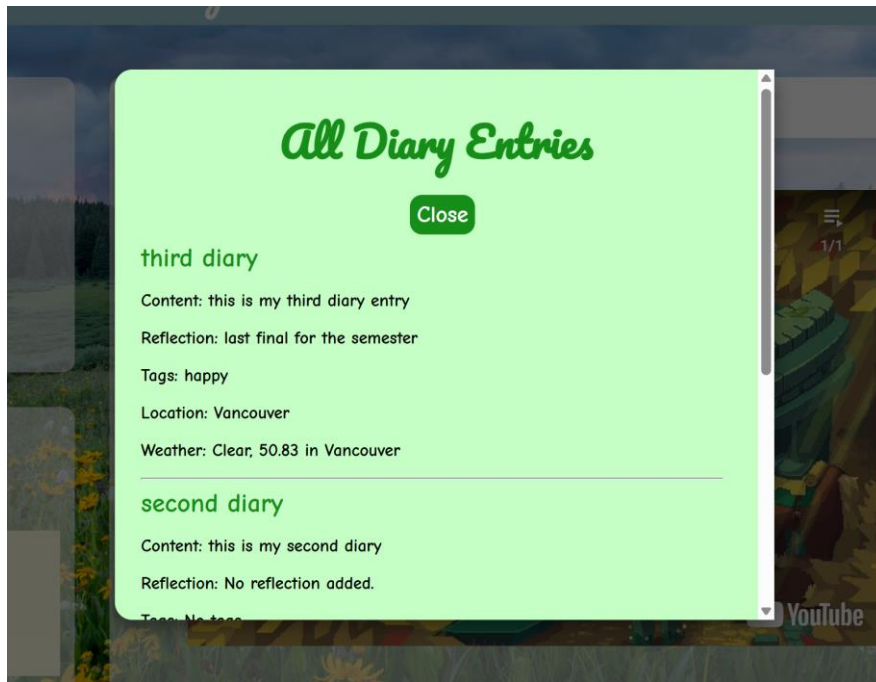


- Can close once done viewing



Viewing ALL entries:

- Click on “Show All Entries”
- Can scroll down and up to view the entries
- Can close once done viewing



second diary

Content: this is my second diary

Reflection: No reflection added.

Tags: No tags.

Location: Texas

Weather: Clouds, 64.35 in Texas

first diary

Content: this is my first diary

Reflection: wow

Tags: happy

Location: Vancouver

Weather: Clear, 50.83 in Vancouver

Close

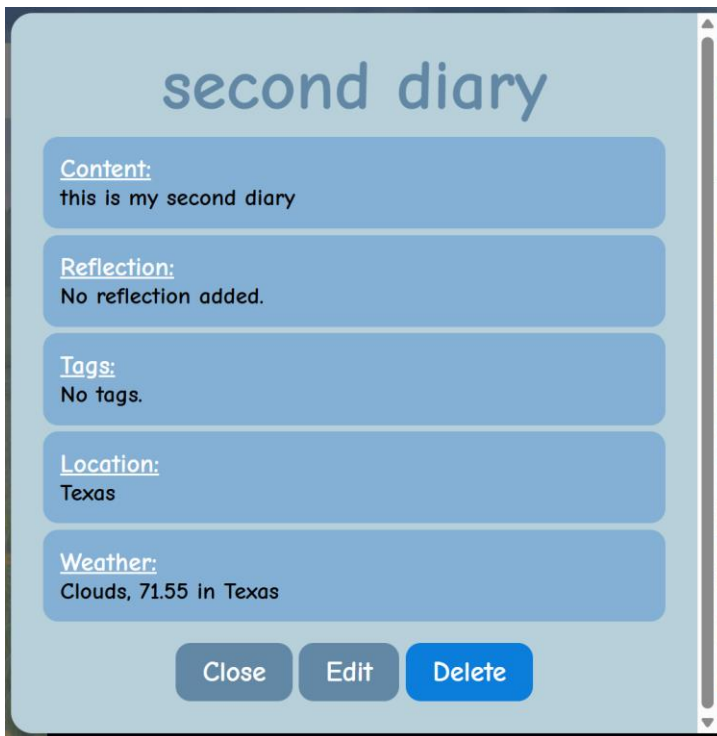
YouTube

Deleting a particular entry:

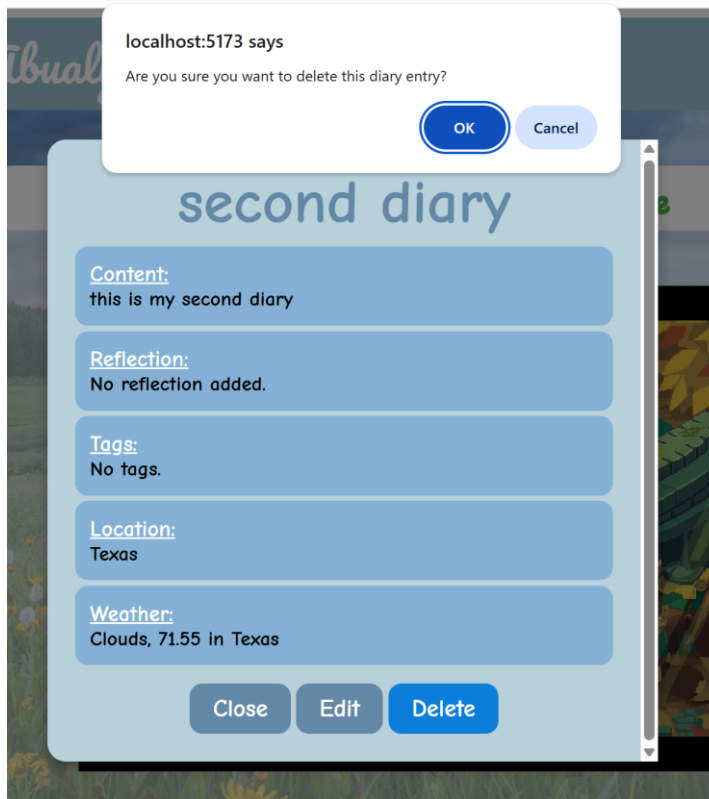
- Click on the entry you want to view from the right side of the screen (will be hovered)



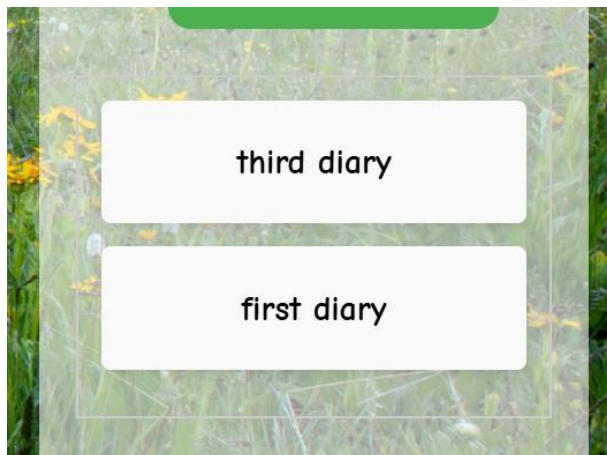
- Click on the “delete” button (will be hovered)



- Pop up will ask again if you are sure you want to delete, click yes if you are sure, no if you are not sure



- At this point, entry should be deleted

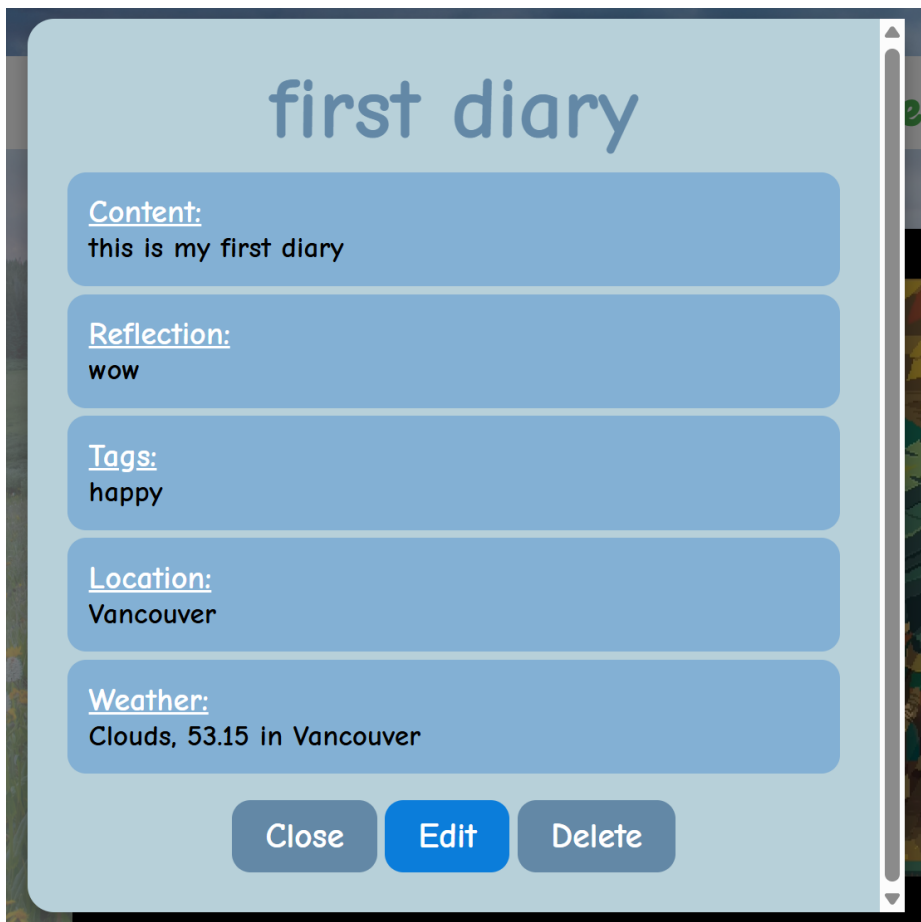


Update diary entry:

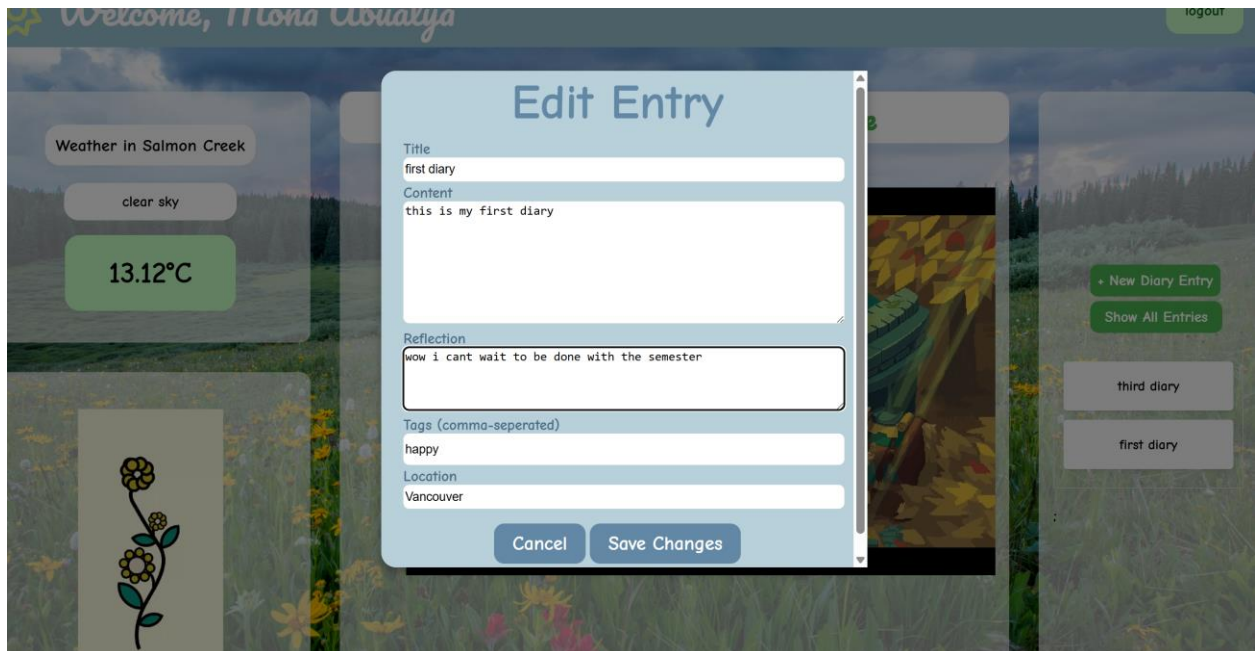
- Click on the entry you want to view from the right side of the screen (will be hovered)



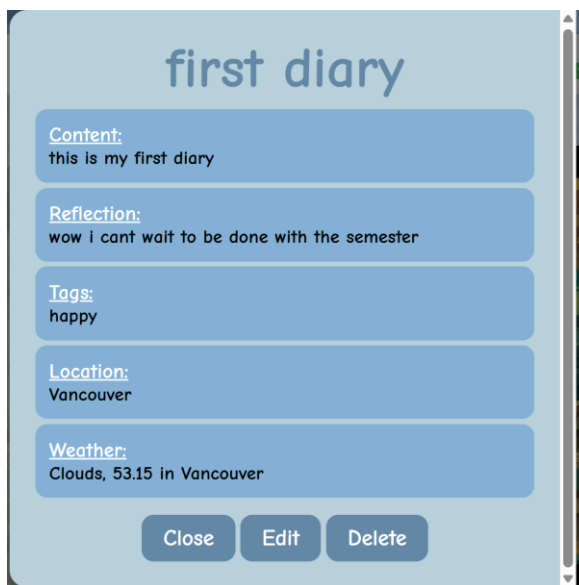
- Click on the “edit” button (will be hovered)



- Update the content of the field you want and then click on “save changes” (updating the location will not update the weather data, unfortunately – frontend issue. But in the backend, weather data will update if location was updated)
- In the below screenshot – I am updating the reflection field



- To check if field was updated, click on the entry you just updated just like the steps above to view a particular entry and you should see that it has been updated
- The below screenshot – reflection updated from “wow” to “wow I cant wait to be done with the semester”



DESCRIPTION OF REACT COMPONENTS HIERARCHY:

The App.jsx defines routes: public <Login> and a protected <Dashboard> that is only accessible to users who have successfully logged in. The login contains the login button component, in which the user can click to log in. The Dashboard is the main authenticated view and is composed of the Header, the WeatherWidget, NewEntryForm, DiaryList, DiaryEntryCard, FlowerAnimation and LofiBeats.

The header is for identification purposes – the name of the authenticated user will show.

WeatherWidget is for contextual weather info – we have it so that it displays your current weather information depending on your current location (you must enable location access for this to display effectively)

NewEntryForm is for diary submission, it initializes the fields that the user will input in

Diary List contains the functionalities of closing, editing, viewing, and deleting. To make the content more interesting and unique, we decided to use modals to display it.

DiaryEntryCard renders a YouTube video with background music for entertainment purposes.

LofiBeats and Flowers Animation are optional UX components that enhance the atmosphere but are not essential for functionality. These are present on the dashboard.

In our main.jsx file initialize the React application to define a top-level hierarchy. It wraps the entire app in a <React.StrictMode> for development safety, <BrowserRouter> for client-side navigation, and <AuthProvider> to manage authentication states. The actual application UI and routing are handled inside the <App /> component.

DESCRIPTION OF ROUTING CONFIGURATION:

In our app, react-router-dom <Routes> and <Route> components are defined.

Authentication is managed using AuthContext, and navigation is controlled based on the user's login status – whether a valid JWT is present.

For the root route (“/”), if the user is authenticated, they are redirected to /dashboard using <Navigate />. If not authenticated, they are shown the <Login /> page.

For the dashboard route (“/dashboard”), if the user is authenticated, they are shown the <Dashboard /> component. If they are not, they are redirected back to the root (Login page)

The purpose is to protect unwanted users from accessing the internal components of the application – only authenticated users can access the Dashboard, and unauthenticated users are always directed to the login screen.